Персональная ЭВМ ПК-01 „Львов“

Описание языка «БЕЙСИК»

00001-01 35 01

1990

УТВЕРЖДЕН

589.0196339.00001-01 35 01-ЛУ

ПЕРСОНАЛЬНАЯ ЭВМ ПК-01 "ЛЬВОВ"

Описание языка БЕЙСИК

589.0196339.00001-01 35 01

ВНИМАНИЕ!

В таблице 1 читать:

Фон	Палитра	Цвет	Должно быть
4	4	3	зел
5	4	3	зел
6	3	1	чер
6	3	2	гол
6	4	1	чер
6	4	2	бел
7	5	1	кр
7	5	2	бел
7	5	3	зел
7	6	1	кр
7	6	2	гол
7	6	3	зел

Зак. 1308—400.

АННОТАЦИЯ

В настоящем документе содержится описание языка БЕЙСИК, правила составления и выполнения программ на языке БЕЙСИК.

В документе описаны основные элементы языка БЕЙСИК: константы, переменные, массивы. Представлены операторы, функции, которые содержит интерпретатор БЕЙСИК, а также способ написания функций, составленных самим пользователем.

Описаны операторы для работы с массивами. Представлены команды для отладки и запуска программных файлов.

Описана система ввода-вывода.

СОДЕРЖАНИЕ

I. ОБЩИЕ СВЕДЕНИЯ И СПОСОБ ОПИСАНИЯ ЯЗЫКА

I.I. Назначение языка

Язык БЕЙСИК является одним из простых в изучении языков программирования. Предназначен для решения вычислительных и планово-экономических задач, проведения инженерных расчетов, разработки тестовых, игровых, учебных программ.

Программа на языке БЕЙСИК представляет собой набор операторов, объединенных в логические блоки.

Диалоговый режим интерпретатора БЕЙСИК облегчает отладку программ. Возможна работа в режиме непосредственного исполнения.

I.2. Способ описания языка

При описании форматов предложений (синтаксиса предложений) языка применяются следующие условные обозначения. Заглавные буквы, знаки пунктуации и другие специальные знаки (за исключением угловых, квадратных и фигурных скобок) изображают конструкции, которые должны быть записаны в таком же виде, как показано в приводимом формате.

Последовательности слов, записанные строчными буквами и заключенные в угловые скобки <и> , определяют синтаксические конструкции, которые записаны где-либо в другой части данного документа. Эти слова в какой-то мере отражают природу и смысл таких конструкций. Если последовательность конструкций заключена в квадратные скобки [и] , то это означает, что данная последовательность является необязательной и может быть опущена. Варианты последовательностей конструкций, заключенные в фигурные скобки {и} , являются альтернативными и должен быть выбран один из этих вариантов.

2. ЭЛЕМЕНТЫ И ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА

2.I. Набор символов

В языке БЕЙСИК ПК-0I используются следующие символы:

1) 26 букв латинского алфавита (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z);

2) 3I буква русского алфавита (А, Б, В, Г, Д, Е, Ж, З, И, К, Й, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Э, Ы, Ю, Я, Ь), твердый знак (Ъ) в тексте отображается символом __ ;

3) I0 десятичных цифр (0, I, 2, 3, 4, 5, 6, 7, 8, 9);

4) знаки арифметических операций:

+ - плюс;

- - минус;

* - звездочка;

/ - наклонная черта (знак деления);

∧ - стрелочка (знак возведения в степень);

5) знаки операций отношения:

< - знак меньше;

> - знак больше;

= - знак равенства;

6) синтаксические знаки:

(- открывающая круглая скобка;

) - закрывающая круглая скобка;

, - запятая;

. - точка;

" - кавычки;

; - точка с запятой;

: - двоеточие;

7) специальные знаки:

¤ - знак денежной единицы (признак символьных данных);

? - вопросительный знак;

␣ - пробел.

Некоторые комбинации символов имеют специальное значение:

<= - меньше или равно;

>= - больше или равно;

<> - не равно.

Употребление слов русского алфавита ограничено по сравнению с английскими буквами (как правило, буквы русского алфавита употребляются только как элементы символьных констант и в операторах комментария).

2.2. Структура программы. Комментарии в программе

В языке программирования БЕЙСИК программа состоит из строк операторов. Каждая строка начинается с номера строки. Номер строки- это целое десятичное число в диапазоне от 0 до 65629. За номером строки располагается служебное слово, определяющее характер действия и текст оператора.

Номера строк служат метками операторов, а также указывают порядок выполнения программы.

При составлении программы рекомендуется нумеровать строки с определенным шагом (например: I0,20,30 и т.д.). Это позволит при отладке вставлять пропущенные или дополнительные строки.

Программа завершается оператором END . Программа также может выполняться без наличия оператора END .

В одной строке может быть записан один или несколько операторов. При записи в одной строке нескольких операторов их следует разделять двоеточием (:). Однако длина строки не должна превышать 254 знаков.

При вводе программы с клавиатуры терминала каждую строку заканчивают нажатием клавиши ВК (возврат каретки).

При вводе или написании программы возникает необходимость в дополнительных пояснениях к программе, предназначенных для объяснения назначения программы, описания назначения ее переменных данных, описания алгоритма и т.п. Для этих целей используется оператор REM.

Формат оператора:

<номер строки> REM <сообщение>

где сообщение может содержать любые печатные знаки клавиатуры.

Сообщение, следующее за словом REM, никаких действий интерпретатора не вызывает.

2.3. Стандартные функции

Для вычисления наиболее встречающихся математических функций в БЕЙСИК включены следующие стандартные функции:

ABS (X) – модуль

SGN (X) – знак X

INT (X) – наибольшее целое число, которое меньше или равно X

$$SGN(x) \begin{cases} -I & \text{при } X<0 \\ 0 & \text{при } X=0 \\ I & \text{при } X>0 \end{cases}$$

COS (X) – косинус X (COS X)

SIN (X) – синус X (SIN X)

TAN (X) – тангенс X ($tg\,X$)

ATN (X) – арктангенс X ($arctg\,X$)

SQR (X) – корень квадратный от X ($\sqrt{X}$)

EXP (X) – экспонента X (e^X)

LOG (X) – натуральный логарифм ($\ln x$)

RND (X) – выборка случайного числа в интервале [0,I] (обращение к генератору случайных чисел).

Аргументом функции может быть любое арифметическое выражение.

Аргументы тригонометрических функций SIN, COS , TAN должны задаваться в радианах. В функции RND (X) необходимо указать фиктивный аргумент, например X.

3. ЭЛЕМЕНТЫ ДАННЫХ

3.I. Числовые и символьные данные

Для того, чтобы интерпретатор мог обеспечивать формирование соответствующих команд для обработки информации, ему должен быть известен тип данных, которые подлежат обработке. В языке БЕЙСИК ПК-0I используются данные двух типов:

I) числовые данные;

2) символьные данные.

Числовые данные представляются десятичными числами – целыми и действительными.

Целое число – это конечная последовательность цифр с возможным знаком плюс или минус перед ней, например:

I986 - 36 + I25

Действительные десятичные числа записываются двумя способами:

- с использованием десятичной точки (в естественной форме) или с использованием латинской буквы E (в нормализованной форме), например:

2,5 IE-I0

-3.75 -36.7E5

25 +25E-4

+I4.6

Диапазон числовых данных $0{,}94.I0^{-38}...I{,}7.I0^{38}$. Разрядность – 6 десятичных цифр.

Символьная константа (строка) – это последовательность символов из набора символов БЕЙСИК ПК-0I, заключенных в кавычки. Например: "BASIC" "2+3" "ЛЬВОВ" "АТИ БАТИ"

Символьная строка может содержать от 0 до n символов ($n \le 255$).

Объем памяти, выделенной под символьные константы (см. раздел [illegible]),

можно определить при помощи функции FRE.

Величины, значения которых в процессе выполнения программы могут меняться, называются переменными. В БЕЙСИКе ПК-0I употребляются два типа переменных - числовые и символьные. Числовая переменная обозначает место в памяти компьютера, в которое будет записано числовое значение. Символьная переменная резервирует место для символьных значений. В БЕЙСИКе употребляются простые переменные и переменные с индексами. Переменные с индексами предназначены для обозначения элементов массивов и описываются дальше.

Числовая переменная обозначается в БЕЙСИКе ПК-0I идентификатором, который состоит из одной или двух латинских букв или латинской буквы и последующей цифры. Например: X, PI, RD, A2.

Переменным присваиваются значения с помощью оператора присваивания LET, операторов INPUT, READ , которые будут описаны далее. Перед выполнением всем переменным присваивается значение 0.

Наряду с числовыми переменными в языке БЕЙСИК ПК-0I используются символьные переменные, которые могут принимать значения символьных констант. Любое имя переменных, за которой следует знак ¤ (знак денежной единицы), указывает символьную переменную.

Пример: A¤ , A2¤ , DM¤ , RG¤

Над символьными переменными и (или) константами могут выполняться следующие операции *отношения*:

=, <>, < , <=, > , >=

При выполнении перечисленных операций сравнения каждая символьная строка представляется в виде целых чисел. Каждому символу отвечает целое число (в соответствии с кодом КОИ-7). При сравнении строк их длины предварительно выравниваются путем приписывания справа к более короткой недостающего числа пробелов.

3.2. Переменные с индексами

Массивом называется конечная совокупность величин, обозначенных одним и тем же идентификатором и поставленных в соответствие упорядоченным набором значений некоторой системы целочисленных параметров, каждый из которых может принимать любое значение, начиная с нуля и кончая некоторым своим наибольшим значением. Элемент массива называется переменной с индексами. В одном массиве могут содержаться величины только одного типа. Поэтому различают массивы числовых данных и массивы символьных данных.

Общий идентификатор всех величин, входящих в состав массива, называется идентификатором массива. Им может быть любое принятое в БЕЙСИКЕ ПК-0I имя переменной. При этом идентификаторы массивов символьных величин оканчиваются знаком денежной единицы (¤).

Индексная переменная обозначается записью, состоящей из идентификатора массива и следующего за ним в круглых скобках списка индексов. Число индексов не должно превышать два и в списке они разделяются запятой. Индекс в общем случае является арифметическим выражением. В момент использования индекс должен иметь положительное целое значение. Индексы элементов массивов принимают значения целых чисел, начиная с Ø, т.е. для одномерного массива элементы нумеруются A(Ø), A(I), A(2) и т.д.; соответственно элементы двумерного массива - B(Ø,Ø), B(Ø,I), B(Ø,2), ..., B(I,Ø), B(I,2) и т.д.

Для каждого массива в программе должен быть указан его максимальный размер, т.е. максимальные значения, которые могут принимать индексы. Это нужно для того, чтобы интерпретатор мог зарезервировать соответствующий объем памяти. Информацию об идентификаторах массивов и их размерностях указывают в операторе DIM, который имеет следующий формат:

<номер строки> DIM $V_1(M_1)$, $V_2(M_2)$,... $V_n(M_n)$

где V_i - идентификатор массива;

M_i - размерность массивов V_i (одно или два целых числа, разделенных запятыми).

Например:

IØ DIM A(IØ), MT (3,4), S¤(6)

Под массив A отводится II элементов, под матрицу MT - 4x5 = 2Ø элементов и символьный массив S¤ -7 элементов. Описание массивов должно предшествовать использованию их элементов в других операторах.

Если в программе размерность индексной переменной не указывается с помощью оператора DIM, то по умолчанию принимается размерность IØ в каждом измерении.

Оператор DIM относится к невыполняемым операторам.

Первоначально все элементы описанных массивов имеют нулевые значения. Для изменения значений индексных переменных используют те же операторы БЕЙСИК ПК-0I, что и в случае простых переменных: оператор присваивания, READ и INPUT.

4. ВЫРАЖЕНИЯ И ОПЕРАТОР ПРИСВАИВАНИЯ

4.I. Арифметические выражения

Арифметические выражения представляют собой произвольную комбинацию идентификаторов, констант и функций, разделенных знаками операций и круглыми скобками так, чтобы образовать имеющее смысл математическое выражение. Допустимы следующие знаки операций:

+ - сложение;

- - вычитание;

* - умножение;

/ - деление;

^ - возведение в степень.

Если в выражении отсутствуют круглые скобки, определяющие порядок выполнения операций, то они выполняются в следующей последовательности:

- вычисление функций;
- возведение в степень;
- умножение и деление;
- сложение и вычитание.

Операции одного ранга выполняются последовательно слева направо (включая и операции возведения в степень).

Над целыми числами могут выполняться логические операции AND , OR, NOT. Эти операции выполняются поразрядно над двоичными представлениями целых чисел в дополнительном коде. Результатом логической операции является целое число, состоящее из последовательности битов, образованных в результате применения данной логической операции к битам заданных операндов. Например:

```
2  OR  3 = 3
2  AND 3 = 2
   NOT 3 = -4
```

Чтобы понять эти результаты, необходимо помнить, что целые десятичные числа в ПК-0I представляются как двоичные числа в дополнительном коде (при этом старший разряд отводится под знак числа).

Если в арифметических выражениях употребляются и арифметические, и логические операции, то при отсутствии скобок логические операции выполняются после арифметических, причем в таком порядке старшинства между собой: NOT, AND, OR. Логические операции одного ранга выполняются слева направо.

4.2. Символьные функции и выражения

Под символьным выражением понимается любая последовательность символьных констант, переменных и функций, соединенных с помощью операции конкатенации (+). Операция конкатенации определяет, что два операнда соединяются вместе и образуют новую строку символов, где за последним символом левого операнда сразу же следует первый символ правого операнда. Длина строки, полученной в результате, равна сумме длин операндов. Например:

"ТРАМ" + "ТАРАРАМ" = "ТРАМТАРАРАМ"

В БЕЙСИКе ПК-0I могут использоваться следующие функции над символьными строками LEFT ¤ (<строка> , N) - выделяет из "строки" подстроку с первого до N-го знака. Здесь и далее над аргументом <строка> будем понимать символьное выражение, а аргумент N (в других функциях NI, N2) должен быть целым числом или выражением, принимающим целочисленное значение. Если N равно или больше длины строки, выдается вся строка. Если N=Ø, выдается пустая строка.
Например: функция LEFT¤("ABCD", 3) дает результат "ABC".
RIGHT¤(<строка> , N) - выделяет наиболее правые символы "строки". Если N равно или превышает длину строки, выдается вся строка. Если N=Ø,выдается пустая строка. Например: функция RIGHT ¤("ABCDE",2) дает строку "DE".

MID¤(<строка>, NI, N2) - проверяет "строку" и возвращает N2 символов, начиная с позиции NI. Если NI больше, чем строка, MID¤ возвращает пустую строку. Например: функция MID¤("ABCDE", 2, 3) дает результат "BCD".

LEN (<строка>) - функция определяет длину строки, то есть возвращает целое число, равное количеству символов в строке.

Например: Функция LEN ("ABCDE") дает результат 5.

VAL (<строка>) - функция преобразует цифровую строку в числовое значение. Строка может включать цифры, знаки "+", "-", "." и "E". Если строка содержит другие знаки, то значение функции равно нулю.

Например: VAL ("IE3") дает числовое значение IØØØ.

STR¤(<выражение>) - действие функции противоположно действию функции VAL : она преобразует числовое значение в цифровую строку.

Например: STR¤(9372) дает строку 9372.

ASC (<строка>) - функция возвращает код КОИ-7 первого символа указанной строки. Функция ASC ("ABCD") дает значение 65.

CHR¤(<выражение>) - функция возвращает символ КОИ-7, код которого равен значению выражения. Выражение должно приводиться к целому числу между Ø и 255.

Например: Функция CHR¤(65) дает символ A.

HEX¤(<выражение>) - функция возвращает строку, значение которой эквивалентно шестнадцатиричному представлению выражения.

Например: Функция HEX ¤(32) дает строку "2Ø".

INKEY¤ - функция (без аргумента), предназначенная для опроса клавиатуры. Функция возвращает пустую строку, если не нажата ни одна клавиша, или строку с символом, соответствующим нажатой клавише.

4.3. Логические выражения

Логическим выражением называют некоторую комбинацию констант и переменных (числовых или символьных), соединенных знаками операций

отношения, логических и арифметических операций и скобками, и принимающие только два значения – "истина" или "ложь" (Ø или I).

Для логических выражений установлен следующий порядок выполнения действий:

I) сначала действует правило скобок;

2) вычисляются значения функций;

3) выполняются арифметические операции, т.е. сначала – возведение в степень, затем умножение и деление и, наконец, сложение и вычитание;

4) выполняются операции отношения (= – равно, < – меньше, <= – меньше или равно, > – больше, >= – больше или равно, <> – не равно);

5) логические операции в следующем порядке:

NOT – логическое отрицание;

AND – конъюнкция;

OR – дизъюнкция.

Операция отношения устанавливает определенные отношения между числовыми или символьными величинами. Отношение имеет формат:
<выражение I (арифметическое или символьное)> <знак операции отношения> <выражение 2 (арифметическое или символьное)>.

Символьное отношение означает последовательное попарное сравнение слева направо символов строк операндов в соответствии с упорядоченностью символов кода КОИ-7.

Результат выполнения операций отношения есть "истина", если выражения удовлетворяют отношению, и "ложь" – в противном случае.

4.4. Оператор присваивания

Оператор присваивания имеет формат:

<номер строки> LET <переменная> = <выражение>

Действие этого оператора состоит в вычислении значения выражения, записанного справа от знака равенства, и присвоении этого зна-

чения переменной, указанной слева от знака равенства.

Например:

```
IØ  LET   A = 6.3
2Ø  LET   B = B+2
3Ø  LET   A$= "BASIC"
```

В операторе присваивания могут употребляться переменные числового и символьного типа, простые и с индексами. Необходимо, чтобы тип переменной совпадал с типом выражения (т.е. числовая переменная и арифметическое выражение, символьная переменная и символьное выражение).

БЕЙСИК ПК-0I позволяет опускать слово LET в операторе.

Например:

```
IØ  A = 6.3
2Ø  B = B+2
3Ø  A$= "BASIC"
```

5. ОПЕРАТОРЫ УПРАВЛЕНИЯ

5.I. Операторы перехода и оператор STOP

В программе на языке БЕЙСИК ПК-0I операторы выполняются в порядке возрастания номеров строк. Изменение порядка выполнения оператора производится с помощью операторов управления. К этой группе операторов относятся операторы перехода, оператор STOP, условные операторы и оператор цикла.

В языке БЕЙСИК ПК-0I существует два вида операторов перехода: оператор безусловного перехода GOTO и оператор перехода по вычислению ON - GOTO.

Оператор безусловного перехода GOTO используется в том случае, когда в программе требуется осуществить безусловный переход к какой-либо другой строке с нарушением естественного порядка выполнения операторов программы, задаваемого номерами строк.

Формат оператора GOTO:

⟨номер строки⟩ GOTO ⟨номер строки I⟩

где

⟨номер строки I⟩ - это номер строки, к которой осуществляется переход. Номер строки, к которой осуществляется переход в программе, может быть либо больше, либо меньше текущего номера строки.

Оператор ON - GOTO позволяет осуществить переход к одной из нескольких указанных строк в зависимости от того, какое значение имеет выражение во время выполнения оператора. Этот оператор имеет следующий формат:

⟨номер строки⟩ ON ⟨ариф. выражение⟩ GOTO ⟨список номеров строк⟩

Во время выполнения оператора вычисляется выражение и целая часть его используется в качестве указателя перехода на один из перечисленных номеров строк.

Например, оператор

5Ø ON P GOTO 1ØØ, I4Ø, I8Ø

передает управление строке IØØ, если P=I, строке I4Ø, если P=2, или строке I8Ø, если P=3.

Любые другие значения P (не выходящие за пределы Ø-255) приводят к передаче управления на следующий оператор.

Оператор STOP вызывает прекращение вычислений по программе (останов программы) и выдачу об этом сообщения на терминал. Формат оператора STOP:

⟨номер строки⟩ STOP

5.2. Условные операторы

Условные операторы позволяют, в зависимости от результата проверки некоторого условия, выполнить или не выполнить некоторый оператор, а также передать управление другой строке. Формат оператора следующий:

⟨номер строки⟩ IF ⟨условие⟩ { THEN ⟨номер строки I⟩ | THEN ⟨оператор⟩ | GOTO ⟨номер строки I⟩ }

Условие представляет собой логическое выражение. В случае выполнения условия, т.е., когда оно принимает истинное значение, то управление передается на оператор с номером строки I (варианты "THEN ⟨номер строки I⟩" и "GOTO ⟨номер строки I⟩") или выполняется оператор, записанный после THEN (вариант "THEN ⟨оператор⟩"); в случае невыполнения условия управление передается оператору с последующим номером строки.

Поскольку любой оператор может следовать за словом THEN, то возможно вложение условных операторов до любого требуемого уровня. Например:

```
2Ø IF X>Y THEN IF Y>Z THEN PRINT "X>Y>Z"
```

Этот оператор эквивалентен следующему:

20 IF X>Y AND Y>Z THEN PRINT "X>Y>Z"

6. ОПЕРАТОР ЦИКЛА FOR-NEXT

Циклические программы (участки программ) можно записывать, используя операторы IF и GOTO. Но для упрощения процедуры составления циклов и с целью их четкого выделения в программе могут быть использованы специальные операторы FOR и NEXT. Оператор FOR, называемый заголовком цикла, всегда предшествует повторяющейся группе операторов, составляющих так называемое тело цикла.

Формат оператора FOR:

<номер строки> FOR <имя переменной> = <выражение I>
TO <выражение 2> [STEP <выражение 3>]

Имя переменной в операторе FOR называют управляющей переменной или параметром цикла, а <выражение I>, <выражение 2> и <выражение 3> представляют собой начальное значение параметра цикла, конечное значение параметра цикла и шаг изменения параметра цикла соответственно. Если " STEP <выражение 3> " опущено, то шаг изменения параметра цикла полагается равным I.

Во время выполнения оператора FOR вычисляются параметры цикла (начальное и конечное значения управляющей переменной, величина шага изменения) и присваивается начальное значение управляющей переменной. Тело цикла начинается вслед за оператором FOR и завершается оператором NEXT, который имеет следующий формат:

<номер строки> NEXT <имя переменной>

Здесь <имя переменной> соответствует управляющей переменной в заголовке цикла. При выполнении оператора NEXT производится изменение значения управляющей переменной на величину шага и производится анализ на конец цикла. Цикл повторяется до тех пор, пока значение управляющей переменной не станет строго больше (при положительном шаге изменения) или строго меньше (при отрицательном шаге изменения) конечного значения. Можно выйти из цикла и в том случае, если параметр цикла не достигает конечного значения. Для этого используют

операторы условного и безусловного перехода.

При использовании вложенных циклов (циклов в цикле) требуется, чтобы область действия внутреннего цикла полностью находилась в области действия внешнего цикла.

Операторы FOR и NEXT не могут выполняться в операторе IF.

7. ФУНКЦИИ

Кроме стандартных функций, предназначенных для вычисления математических выражений, в языке БЕЙСИК ПК-0I допускается употребление функций, определяемых пользователем. Если в программе необходимо несколько раз вычислить одно и то же выражение при различных значениях некоторого параметра, то целесообразно создать "свою" функцию.

Описать функцию можно с помощью определяющего оператора DEF, формат которого следующий:

⟨номер строки⟩ DEF FNα (⟨аргумент⟩) = ⟨выражение⟩

где

FNα – имя функции, состоящее из обязательных букв FN и имени произвольной переменной.

Пример:

```
IØ  DEF FNA(X) = SIN(X)^2 -COS(X)^2
```

Аргумент употребляемый в описании функции (в операторе DEF), называют формальным параметром функции. Выражение, записываемое в правой части определения функции, может быть произвольным арифметическим выражением, зависящим, как правило, от формального параметра. Однако, оно не обязательно должно зависеть от аргумента и может содержать также другие переменные, определенные в программе, которые не являются аргументом, например:

```
2Ø  DEF  FN A(X) = (X-A)*B + TAN(C/X)
```

Если в программе введено описание функции, то можно обращаться к нему, т.е. употреблять обозначение этой функции (иначе – указатель функции) в различных операторах, выражениях и т.п. В указателе функции все формальные параметры должны быть заменены на фактические, например, в программе:

```
IØ  DEF  FNA(X) = X^2 + SIN(X)
2Ø  A = FNA(2.5) + 2*FNA(3)
```

Формальный параметр X вначале заменяется на 2,5, а затем на 3.

Фактические параметры могут быть любыми арифметическими выражениями. При обращении к функции сначала вычисляется значение фактического параметра, далее он подставляется вместо формального параметра в описание функции, и вычисляется значения выражения, записанного в правой части определения функции. Это значение является значением указателя функции при данных фактических параметрах.

Наконец, отметим, что оператор DEF является неисполнимым оператором и поэтому может размещаться в любой части программы, например, в начале.

8. ПОДПРОГРАММЫ

8.I. Операторы GOSUB и RETURN

Последовательность повторяющихся в программе операторов может быть оформлена в виде подпрограммы. Подпрограмма на языке БЕЙСИК ПК-0I - это часть программы, которая реализует операции, выполнения которых требуется в нескольких точках программы.

Для перехода к подпрограмме используется оператор, имеющий следующий формат:

<номер строки> GOSUB <номер строки подпрограммы>

где

<номер строки подпрограммы> - номер строки подпрограммы, содержащий оператор подпрограммы, с которого начинается ее выполнение (точка входа). При выполнении оператора GOSUB управление передается в точку входа в подпрограмму.

Первая строка подпрограммы может начинаться оператором комментария (оператором REM) или любым выполняемым оператором.

Подпрограмма обрабатывается интерпретатором до тех пор, пока не встретится оператор RETURN, формат которого следующий:

<номер строки> RETURN

Оператор RETURN обуславливает возврат управления на оператор, следующий за оператором GOSUB.

БЕЙСИК ПК-0I допускает использование в одной программе нескольких подпрограмм. Подпрограммы могут быть вложенными, то есть одна подпрограмма может осуществить вызов другой подпрограммы. Если при выполнении подпрограммы встречается оператор RETURN, то он вызывает передачу управления тому оператору, который следует после оператора GOSUB, произведшего обращение к этой подпрограмме (в точку возврата).

Программа может содержать несколько точек входа, кроме того, подпрограмма может иметь более одного оператора RETURN, причем, как только встречается первый из них, осуществляется выход из подпрограммы.

8.2. Оператор ON - GOSUB

Оператор ON - GOSUB используется (аналогично оператору ON - GOTO) для условной передачи управления одной из нескольких подпрограмм или к одной из нескольких входных точек одной (или нескольких) подпрограмм. Формат оператора

<номер строки> ON <арифметическое выражение> GOSUB <список номеров строк подпрограммы>

Во время выполнения оператора ON - GOSUB вычисляется значение арифметического выражения и целая часть этого значения (n) используется в качестве указателя на один из перечисленных номеров строк в списке. В результате происходит передача управления подпрограмме, которая начинается номером строки, находящимся в списке на n -ом месте. Оператор RETURN возвращает управление оператору, следующему за оператором ON - GOSUB.

9. СРЕДСТВА МАШИННОЙ ГРАФИКИ И УПРАВЛЕНИЯ ЗВУКОМ

9.1. Формирование изображения на экране терминала

Используя средства машинной графики языка БЕЙСИК версии 2.0 ПК-01, на экране терминала можно создавать произвольные цветные графические изображения, элементами которых являются точки, линии и прямоугольники. Выводимое на экран изображение обрамлено рамкой (окантовкой), которая, как правило, включает часть телевизионной развертки со значительными нелинейными искажениями. Внутри этой рамки находится поле изображения размером 200 х 222 элемента (пиксела).

После очистки экрана это поле окрашено в цвет фона. Элементы изображения выводятся одним из трех цветов переднего плана из палитры 8 цветов.

Очистка экрана, т.е. закрашивание всех пикселов поля изображения в цвет фона, осуществляется оператором CLS. Его формат:

<номер строки> CLS

Задание конкретного цвета переднего плана, цвета фона и необходимой палитры цветов осуществляется оператором COLOR. Формат этого оператора:

<номер строки> COLOR [<цвет>] [, <фон>] [, <палитра>]]

где <цвет> - номер цвета переднего плана. Параметр должен принимать значения от Ø до 3. Значение Ø задает цвет фона. Этот параметр задает цвет, которым будет выводиться текстовая информация, а также графическая информация, если в графических операторах цвет переднего плана не был установлен;

<фон> - номер цвета фона. Параметр должен принимать значения в диапазоне от Ø до 7;

<палитра>- номер палитры. Параметр должен принимать значения в диапазоне от Ø до 6.

Параметры <цвет>, <фон> и <палитра> кодируются в соответствии с таблицей I.

Таблица I

ФОН / номер фона	Номер палитры Ø	I	2	3	4	5	6	Номер цвета переднего плана
чер Ø	чер	чер	чер	чер	чер	чер	чер	Ø
	зел	зел	зел	зел	гол	гол	бел	I
	син	сир	син	сир	кр	кр	кр	2
	кр	кр	жел	жел	сир	бел	син	3
кр I	кр	кр	кр	кр	кр	кр	кр	Ø
	зел	зел	жел	жел	гол	гол	бел	I
	син	сир	сир	син	чер	чер	чер	2
	жел	жел	чер	чер	сир	бел	син	3
син 2	син	син	син	син	син	син	син	Ø
	зел	зел	жел	гол	гол	гол	гол	I
	сир	сир	сир	чер	кр	чер	кр	2
	кр	жел	чер	сир	сир	бел	бел	3
сир 3	сир	сир	сир	сир	сир	сир	сир	Ø
	зел	зел	жел	гол	гол	бел	бел	I
	син	син	син	чер	кр	кр	чер	2
	кр	жел	чер	бел	бел	син	син	3
бел 4	бел	бел	бел	бел	бел	бел	бел	Ø
	сир	сир	син	син	кр	кр	чер	I
	жел	зел	зел	жел	гол	гол	гол	2
	гол	гол	сир	сир	жел	чер	жел	3

Продолжение табл. I

ФОН / номер фона	Номер палитры Ø	I	2	3	4	5	6	Номер цвета переднего плана
гол 5	гол	гол	гол	гол	гол	гол	гол	Ø
	сир	сир	син	син	кр	кр	чер	I
	жел	зел	зел	жел	бел	бел	бел	2
	син	син	бел	бел	жел	чер	жел	3
зел 6	зел	зел	зел	зел	зел	зел	зел	Ø
	сир	сир	син	кр	кр	кр	кр	I
	жел	жел	жел	бел	гол	бел	гол	2
	гол	син	бел	жел	жел	чер	чер	3
жел 7	жел	жел	жел	жел	жел	зел	зел	Ø
	сир	сир	син	кр	кр	чер	чер	I
	зел	зел	зел	бел	гол	гол	бел	2
	гол	син	бел	чер	чер	жел	жел	3

Примечания: I. Для фона номер 7 и палитр 5 и 6 фон будет зеленого цвета.

2. В таблице применяются такие сокращения:

чер	– черный	бел	– белый
кр	– красный	гол	– голубой
син	– синий	зел	– зеленый
сир	– сиреневый	жел	– желтый

Все параметры должны быть выражениями числового типа; если значение выражения нецелое, то используется только целая часть значения. Неуказанные параметры по умолчанию принимают ранее заданные значения. Если значения параметров выходят за указанные пределы, то

для задания параметров используются младшие разряды их двоичных представлений.

Цвет окантовки экрана соответствует второму номеру цвета переднего плана.

Примеры:

1Ø COLOR I,Ø,Ø - устанавливает зеленый цвет переднего плана для вывода текстовой и графической информации на черном фоне (цвет окантовки синий).

5Ø COLOR 3,2,4 - задает сиреневый цвет переднего плана на синем фоне (цвет окантовки - красный).

95 COLOR , I - меняет только номер цвета фона; номер цвета переднего плана и номер палитры остаются прежними.

9.2. Графические операторы

Графические операторы языка БЕЙСИК 2.0 ПК-0I включают операторы PSET, PRESET, LINE. При выводе изображений при помощи этих операторов адреса пикселов задаются координатами X и У. Пиксел, находящийся в левом верхнем углу поля изображения, имеет координаты (Ø, Ø), в левом нижнем углу - координаты (Ø, 22I), в правом верхнем углу - координаты (I99, Ø) и т.д.

Оператор PSET позволяет вывести точку определенного цвета в заданную точку экрана. Формат оператора:

⟨номер строки⟩ PSET (⟨X - коорд⟩ , ⟨У - коорд⟩) [, ⟨цвет⟩]

где ⟨X - коорд⟩ и ⟨У - коорд⟩ - соответственно координата X и координата У выводимой точки;

⟨цвет⟩ - номер цвета переднего плана, который в совокупности с ранее определенными номером фона и номером палитры определяет цвет выводимой точки.

Все параметры - это выражения числового типа. Если параметр ⟨цвет⟩ опущен, то принимается значение заданное оператором COLOR или

заданное в одном из графических операторов, выполненных раньше.

Примеры:

I00 PSET (5Ø , IØØ), I

25Ø PSET (X, У+I)

Оператор PRESET закрашивает заданную точку экрана в цвет фона, т.е. стирает точку. Формат оператора:

⟨номер строки⟩ PRESET (⟨X - коорд⟩ , ⟨У - коорд⟩)

Назначение и формат параметров ⟨X - коорд⟩ , ⟨У - коорд⟩ такой-же как и у оператора PSET.

Оператор LINE дает возможность выводить на экран отрезки прямых линий, с его помощью можно чертить и закрашивать прямоугольники, стороны которых параллельны сторонам рамки экрана. Формат оператора:

⟨номер строки⟩ LINE [(⟨X - нач⟩ , ⟨У - нач⟩)] - (⟨X - кон⟩ , ⟨У - кон⟩) [, [⟨цвет⟩] [, B [F]]]

где

⟨X - нач⟩, ⟨У - нач⟩ - соответственно координата X и координата У начала отрезка;

⟨X - кон⟩, ⟨У - кон⟩ - соответственно координата X и координата У конца отрезка;

⟨цвет⟩ - номер цвета переднего плана, которым чертится линия.

Необязательный параметр B указывает, что на экран выводится не отрезок прямой линии, а прямоугольник, стороны которого параллельны сторонам окантовки экрана, а координаты крайних точек любых из диагоналей равны ⟨X - нач⟩, ⟨У - нач⟩ и ⟨X - кон⟩, ⟨У - кон⟩.

Необязательный параметр BF предписывает вывод закрашенного прямоугольника.

Формат параметров ⟨X - нач⟩, ⟨У - нач⟩, ⟨X - кон⟩, ⟨У - кон⟩ и цвет такой же как и в операторе PSET.

Если параметр (⟨X - нач⟩, ⟨У - нач⟩) опущен, то в качестве координат начальной точки берутся координаты последней, выведенной предыдущими графическими операторами, точки.

Примеры:

```
IØØ  LINE  (IØ, IØ) - (3Ø, 3Ø),3
IIØ  LINE  - (IØ, 3Ø)
2ØØ  LINE  (5Ø, IØ) - (7Ø, 2Ø),, B
2IØ  LINE  (XI, YI) - (X2, Y2), 2, BF
```

Если понадобится какая-либо обработка графического изображения, то номер цвета конкретной точки можно узнать при помощи функции POINT.

Формат функции:

POINT (⟨X - коорд⟩ , ⟨У - коорд⟩)

Здесь параметры ⟨X - коорд⟩ и ⟨У - коорд⟩ имеют то же назначение и формат, что и в операторах описанных выше.

Для обеспечения преемственности с ранее написанными программами, в интерпретатор БЕЙСИК 2.0 ПК-0I включены также операторы PLOT и DRAW предыдущей версии интерпретатора. Для этих операторов точка начала координат расположена в левом нижнем углу и расположение осей координат соответствует первому квадранту декартовой системы координат.

Оператор PLOT позволяет вывести точку определенного цвета в заданную точку экрана.

Формат оператора:

⟨номер строки⟩ PLOT ⟨X - коорд⟩, ⟨У - коорд⟩, ⟨цвет⟩

Назначение и формат параметров оператора PLOT такой же, как и у оператора PSET.

Оператор DRAW позволяет вывести на экран отрезок прямой линии, который соединяет последний выведенный предыдущим оператором PLOT или DRAW пиксел и точку с координатами X и У, заданными в данном операторе. Цвет пикселов этого отрезка такой же, как и у пиксела, являющегося началом отрезка.

Формат оператора:

⟨номер строки⟩ DRAW ⟨X - коорд⟩ , ⟨У - коорд⟩

Операторы PLOT и DRAW не рекомендуется применять при разработке новых программ, так как из последующих версий они будут исключены.

9.3. Средства управления звуком

Средства воспроизведения звука включают операторы BEEP и SOUND.

Оператор BEEP позволяет выдать короткий звуковой сигнал из динамика или капсуля.

Формат оператора:

⟨номер строки⟩ BEEP

Оператор SOUND позволяет воспроизвести при помощи динамика (капсуля) несложную мелодию.

Формат оператора для воспроизводства отдельного звука:

⟨номер строки⟩ SOUND ⟨код частоты⟩, ⟨код длительности⟩

Формат для последовательности звуков:

⟨номер строки⟩ SOUND ⟨код частоты⟩, ⟨код длительности⟩ ;

...; ⟨код частоты⟩ , ⟨код длительности⟩

где

⟨код частоты⟩ и ⟨код длительности⟩ - числовые выражения.

Код частоты для нот гармонического звукоряда приведены в таблице 2.

Длительность одной единицы значения параметра ⟨код длительности⟩ равна приблизительно 7 мс.

Если код частоты равен Ø, то задается пауза.

Пример: В ниже приведенных операторах закодированы первые два такта мелодии популярной детской песни "Пусть всегда будет солнце".

```
IØ  SOUND  69, 8Ø;  66, 4Ø;  69, 4Ø;  52, I6Ø
2Ø  SOUND  46, 8Ø;  43, 4Ø;  46, 4Ø;  69, I6Ø
```

Таблица 2

Ноты	Малая октава	Первая октава	Вторая октава	Третья октава
до	I56	78	38	I9
до #	I47	73	36	I8
ре	I39	69	34	I7
ре #	I3I	66	32	I6
ми	I23	6I	30	I5
фа	II6	58	29	I4
фа #	II0	55	27	I3
соль	I04	52	26	I2
соль #	98	49	25	
ля	92	46	23	
ля #	87	43	2I	
си	83	4I	20	

I0. СРЕДСТВА РАБОТЫ НА УРОВНЕ МАШИННЫХ КОМАНД

Для квалифицированных пользователей, знакомых с программированием в кодах микропроцессора К580ИК80, в состав интерпретатора включены средства, позволяющие из БЕЙСИК - программы вызывать подпрограммы, составленные в кодах машины и читать, и записывать информацию в конкретные ячейки памяти. К этим средствам относятся операторы POKE, VPOKE, OUT, WAIT и функции PEEK, VPEEK, INP, USR и VAEPTR.

Оператор POKE помещает указанное значение в указанную ячейку памяти.

Формат оператора:

⟨номер строки⟩ POKE ⟨адрес памяти⟩, ⟨значение⟩

где ⟨адрес памяти⟩ - это арифметическое выражение, целая часть которого определяет адрес ячейки памяти, в которую производится запись;

⟨значение⟩ - арифметическое выражение в диапазоне от Ø до 255, целая часть которого записывается в указанную ячейку.

Например:

IØ POKE I2247,35

Функция PEEK читает I байт из указанной ячейки памяти. Формат функции:

PEEK (⟨адрес памяти⟩)

где ⟨адрес памяти⟩ - это арифметическое выражение, целая часть которого определяет адрес ячейки памяти, из которой производится чтение информации.

Например:

2Ø PRINT PEEK (I2247)

Функция USR вызывает подпрограмму, написанную в кодах машины. Адрес точки входа в программу в кодах машины должен быть перед ис-

пользованием функции USR помещен в фиксированные ячейки памяти. В ячейку с адресом 73 помещается младший байт адреса подпрограммы, в ячейку о адресом 74 - старший байт. Если выполнение подпрограммы в кодах машины завершается командой возврат (RET, RZ и т.п.), то происходит возврат в интерпретатор и возобновится выполнение БЕЙСИК-программы.

Формат функции USR:

USR (фиктивный аргумент)

где фиктивный аргумент - это произвольная числовая переменная.

Функция USR возвращает значение Ø.

Например:

Пусть необходимо выполнить подпрограмму в кодах машины, размещенную в памяти, начиная с адреса I2288. Число I2288 в шестнадцатиричной системе счисления представляется как 3ØØØ. Старший байт адреса равен 3Ø (48 в десятиричной), младший байт-ØØ. Вызов этой подпрограммы можно осуществить при помощи следующих операторов:

```
IØ   POKE   73,ØØ
2Ø   POKE   74,48
3Ø   X = USR (X)
```

Оператор VPOKE помещает указанное значение в указанную ячейку видеопамяти.

Формат оператора:

<номер строки> VPOKE <адрес>, <значение>

где <адрес> - это арифметическое выражение, целая часть которого определяет адрес ячейки видеопамяти, в которую производится запись. Значение адреса должно находиться в пределах от Ø до I6383;

<значение> - арифметическое выражение в диапазоне от Ø до 255, целая часть которого записывается в указанную ячейку и кодирует цвет 4-х рядом расположенных пикселов экрана.

Функция VPEEK читает один байт из указанной ячейки видеопамяти.

Формат функции: VPEEK (<адрес>)

Параметр <адрес> задается так-же, как и в операторе VPOKE.

Функция VAEPTR возвращает адрес памяти, по которому хранится значение указанной переменной.

Формат функции:

VAEPTR (<переменная>)

где <переменная> - идентификатор переменной.

Оператор OUT позволяет вывести байт данных на заданный порт вывода.

Формат оператора:

<номер строки> OUT <порт>, <данные>

где <порт> - это адрес порта вывода, на который выводятся данные. Параметры <порт> и <данные> - это выражение, которое должно находиться в пределах от I92 до 255.

Оператор WAIT устанавливает состояние ожидания до тех пор, пока вводимые с порта данные не достигнут определенного значения.

Формат оператора:

<номер строки> WAIT <порт>, <маска AND> [, <маска XOR>]

где <порт> - см. оператор OUT;

<маска AND> и <маска XOR> - выражения, значения которых должно находиться в пределах от Ø до 255. Этот оператор вводит байт данных из указанного порта ввода. С введенными данными и параметром <маска AND> выполняется поразрядная операция "И", а с полученным результатом и параметром <маска XOR> поразрядная операция "исключающее ИЛИ". Процесс ввода и обработки байта данных повторяется до тех пор, пока полученное значение не станет нулевым.

Функция INP позволяет вводить данные с заданного порта ввода.

Формат функции:

INP (<порт>),

где <порт> см. оператор OUT.

11. РАБОТА В РЕЖИМЕ НЕПОСРЕДСТВЕННОГО ИСПОЛНЕНИЯ

11.1. Режим калькулятора

Если в операторе БЕЙСИК ПК-01 опустить номер строки, то такой оператор выполняется немедленно. Это свойство можно использовать для выполнения несложных вычислений.

Пример:

Ввод с терминала строки

PRINT 2,5 * LOG(3,5)

вызовет вывод на экран числа

3.13191

Вместо служебного слова PRINT разрешается использовать вопросительный знак (?). В режиме непосредственного исполнения можно также использовать ряд других операторов (LET, POKE, PLOT, DRAW и др.). Например, при непосредственном выполнении операторов

```
X = 25
Y = SQR (x)
PRINT X
```

будет напечатано:

5

Логически не имеет смысла непосредственное исполнение операторов DEF, DIM, DATA, RESTORE, FOR, NEXT, END, REM, RETURN, STOP; недопустимо использование в режиме непосредственного использования операторов DEF, INPUT, DATA.

11.2. Использование режима непосредственного исполнения при отладке программ

Для облегчения отладки программ программист может разместить в программе операторы STOP. Каждый оператор STOP вызывает останов в выполнении программы, печать номера строки, на которой произошла остановка. В это время программист может проверить различные значения, возможно, изменить их в режиме непосредственного исполнения.

Для продолжения выполнения программы следует ввести директиву CONT (см.раздел 13).

Во время отладки, используя функцию FRE, можно определить объём свободной памяти, не занятой программой или объём свободной памяти, отведенной под размещение строковых данных. Формат функции:

FRE (⟨фиктивный аргумент⟩)

Если задан фиктивный аргумент числового типа, то выдаётся оставшееся количество байт памяти, отводящейся под программу, а если строкового, то под размещение строк.

Пример:

PRINT FRE (X)

12. ОПЕРАТОРЫ ВВОДА - ВЫВОДА

12.1. Операторы READ , DATA и RESTORE

Для ввода данных в программу используются операторы READ и DATA. Оператор DATA содержит значения, которые присваиваются переменным, записанным в операторе READ.

Формат оператора DATA:

⟨номер строки⟩ DATA ⟨список значений⟩

Список значений может включать в себя числовые и символьные данные. Элементы данных в списке должны разделяться запятыми.

Пример:

```
1Ø DATA    I, -2.63, I.5E-6
2Ø DATA    ABCE , 3.5, -26
```

Элементы символьных данных необходимо заключить в кавычки тогда, когда символьная строка включает запятую или пробел.

Пример:

```
1Ø DATA   "AB, CD "
```

В одной программе может быть один или несколько операторов DATA. Списки всех операторов рассматриваются как один блок данных.

Во время выполнения программы информация из блока данных извлекается для присвоения значения переменным с помощью оператора READ . Оператор READ имеет следующий формат:

⟨номер строки⟩ READ ⟨список переменных⟩.

Список переменных представляет собой перечень имен переменных (числовых и символьных, простых и с индексом). Имена переменных в списке разделяются запятыми.

Пример:

```
3Ø READ  M, A, K, BK, SΩ, A2
```

Если при выполнении программы встречается оператор READ , то первой переменной списка оператора READ присваивается первое значение из блока данных, второй переменной - второе значение и т.п.

При этом интерпретатор запоминает последнее значение, присвоенное переменной из списка оператора READ . Поэтому, когда в программе встречается ещё один оператор READ , то переменной присваивается следующее имеющееся значение из блока данных.

Если список значений в блоке данных исчерпался раньше списка переменных в операторе READ , то происходит программное прерывание, а на печать выдается сообщение об ошибке.

Если одни и те же данные необходимо использовать несколько раз, то используются оператор RESTORE (восстановить). Оператор RESTORE в программе должен размещаться перед каждым следующим считыванием данных. Формат оператора RESTORE:

⟨номер строки⟩ RESTORE

⟨номер строки⟩ RESTORE ⟨номер строки I⟩

После его выполнения выборка значений операторами READ начнётся повторно с самого первого значения элемента блока данных, в случае первого формата оператора RESTORE, или со значения элемента блока данных, помещенного в операторе DATA с ⟨номером строки I⟩, во втором случае. Оператор RESTORE можно употреблять в любой момент рабочей программы, не дожидаясь полного исчерпания блока данных.

I2.2. Оператор INPUT

При выполнении некоторых задач исследовательского или игрового характера возникает необходимость корректировать входные данные по ходу счета в зависимости от получающихся результатов. Для этой цели используют оператор INPUT, формат которого следующий:

⟨номер строки⟩ INPUT ⟨список⟩

Список включает в себя имена переменных (численных и символьных). Выполняя оператор INPUT, компьютер делает паузу во время выполнения программы, печатает на терминале знак вопроса "?". Пользователь должен набрать на клавиатуре значения переменных, разделяя их запятыми. Количество и тип значений должны соответствовать количеству и типу переменных в списке. Для ввода набранных значений следует нажать клавишу ВК. Если вводимые символьные данные содержат запятую, то их нужно заключить в кавычки.

Выдача на терминал вопросительного знака (?), при наличии в программе нескольких операторов INPUT, не даёт возможности определить, значения каких именно переменных должен ввести пользователь. Для устранения этого недостатка используют оператор INPUT следующего формата:

⟨номер строки⟩ INPUT "⟨сообщение⟩"; ⟨список⟩

где ⟨сообщение⟩ – произвольная последовательность печатных символов.

Пример:

IØ INPUT "МАССА" ; М

При выполнении этого оператора на терминал выводится

МАССА ?

Пользователь должен набрать значение переменной М и нажать клавишу ВК.

I2.3. Оператор PRINT

Вывод на терминал результатов вычислений и пояснительных текстов осуществляется с помощью оператора PRINT. Формат оператора:

⟨номер строки⟩ PRINT [⟨список⟩]

Элементами списка могут быть числа, переменные, выражения и символьные строки. Оператор PRINT без списка используется для перевода строки.

Длина строки терминала ПК-01 составляет 32 позиции. Эта строка разбита на 2 зоны по I6 позиций. При выводе на печатающее устройство печатная строка разбивается на соответствующее число зон по I6 позиций (5 зон при длине 80 позиций, 8 зон при длине I28 позиций и т.п.). Если за элементом из списка оператора PRINT следует запятая (,), то значение следующего элемента списка будет напечатано в первой свободной зоне, данной или следующей строки.

Если же последним элементом оператора PRINT следует запятая, следующее значение, которого должно выводиться на печать последующим оператором PRINT, будет печататься в следующей зоне печати. Две запятые, стоящие рядом в операторе PRINT, обуславливают пропуск зоны печати.

При необходимости более тесного размещения напечатанных значений, вместо запятой следует использовать точку с запятой (;). При наличии точки с запятой следующее значение печатается через одну позицию справа от предыдущего. Если за последним элементом оператора PRINT следует точка с запятой, следующее значение, которое должно выводиться на печать последующим оператором PRINT, будет печататься через одну позицию справа от предыдущей - на той же строке (если в строке есть место для печати).

12.4. Дополнительные возможности оператора PRINT

Дополнительные удобства для размещения выводимой информации могут быть получены при использовании в операторе PRINT функции TAB(X) и SPC (X). Функция TAB(X) вызывает перемещение курсора (указывающего позицию вывода очередных данных) к позиции с номером X. Позиции в строке нумеруются, начиная с Ø. Формат функции TAB:

TAB (⟨выражение⟩)

Пример:

```
IO PRINT      TAB(5);  X; TAB(25); У
```

Функция SPC (X) возвращает строку пробелов длиной INT (X), которая "вставляется" в выводимую строку. Формат функции SPC:

SPC (⟨выражение⟩)

Пример:

```
IØ PRINT    TAB(5); X; SPC(II); У
```

Иногда при выводе некоторых результатов на печать возникает необходимость в уточнении положения курсора в выводимой строке. Для этого можно воспользовать функцией POS (X), где X-фиктивный аргумент. Эта функция возвращает позицию курсора после последнего

оператора PRINT.

Пример:

```
IØ PRINT "###";
I5 P=POS  (Ø)
```

Переменная P примет значение, равное 4.

Номер строки, в которой находится курсор, можно определить при помощи функции CSRLIN. Эта функция не имеет аргумента и возвращает значение текущей строки курсора в пределах от Ø до 23 (Ø - для верхней строки, 23 - для нижней строки экрана).

12.5. Управление курсором

Для позицирования выводимого на экран терминала текста в БЕЙСИКе 2.0 ПК-01 используется оператор LOCATE, позволяющий переместить курсор в определенное место экрана. Формат оператора LOCATE

⟨номер строки⟩ LOCATE [⟨номер столбца⟩],
[⟨номер строки⟩] [,⟨видимость⟩]

где ⟨номер столбца⟩ - указывает номер позиции в строке, в которую помещается курсор; может принимать значение от Ø до 31;

⟨номер строки⟩ - указывает номер строки, в которую помещается курсор, может принимать значение от Ø до 23;

⟨видимость⟩ - указывает режим вывода курсора; значение I указывает, что курсор выводится, Ø - не выводится.

Если какой-либо параметр не указан, то по умолчанию принимается его предыдущее значение.

13. СРЕДСТАВА ПОДГОТОВКИ И ОТЛАДКИ ПРОГРАММ

13.1. Директивы интерпретатора и средства отладки программы

Директивы интерпретатора предназначены для управления его работой. При вводе директивы, в отличие от оператора, номер строки не ставится. Все директивы выполняются немедленно.

Директива NEW используется перед вводом новой программы. Если в памяти находится старая программа, то она стирается. При выполнении директивы NEW инициализируется ряд системных переменных интерпретатора.

Формат директивы NEW .

NEW

Директива LIST предназначена для получения на экране терминала текста всей или части программы, находящейся в оперативной памяти.

Формат директивы:

LIST [n1] [-n2]

где n1 - номер строки с меньшим номером;

n2 - номер строки с большим номером.

При вводе директивы LIST без параметров, выдается листинг всей программы; с параметром n1 - текст от оператора с номером n1 до конца программы; а в случае указания n1 и n2 - текст программы от оператора с номером строки n1 до оператора с номером строки n2.

При выдаче на экран листинга программы возможен приостанов его выдачи. Для этого необходимо нажать функциональную клавишу $\langle {F5 \atop AS} \rangle$. Повторное нажатие на клавишу $\langle {F5 \atop AS} \rangle$ вызывает выдачу следующей строки программы.

После этого возможно либо продолжение выдачи листинга в автоматическом режиме (путем нажатия на клавишу "пробел"), либо прекращение выдачи листинга и выход в режим диалога (путем нажатия функциональной клавиши FØ - клавиши в верхнем ряду).

Директива RUN предназначена для запуска программы на выполнение.

Формат директивы:

RUN [n]

где n - номер строки оператора, с которого начинается интерпретация программы. При отсутствии параметра n интерпретация начинается с оператора о наименьшим номером строки.

После запуска программы на выполнение возможна ее приостановка, путем нажатия на клавишу $\langle {F5 \atop AS} \rangle$. Повторное нажатие на клавишу $\langle {F5 \atop AS} \rangle$ вызывает выполнение очередного оператора БЕЙСИК - программы, т.е. пошаговое ее выполнение. Возобновление выполнения программы осуществляется путем нажатия на клавишу "пробел". После приостановки программы возможно также прекращение ее выполнения и выход в режим диалога. Такой останов осуществляется путем нажатия на функциональную клавишу FØ (в верхнем ряду клавиши).

Директива CONT предназначена для возобновления программы, остановленной при выполнении оператора STOP, оператора END или при нажатии на функциональную клавишу FØ. Выполнение программы возобновляется с оператора, следующего за оператором, в котором произошел останов.

Формат директивы:

CONT

Директива CLEAR предназначена для очистки всех переменных и массивов в БЕЙСИК-программе и для изменения размеров памяти, предназначенной для хранения строковых переменных.

Формат директивы:

CLEAR [N]

где [N] - целое число, определяющее количество байт памяти, отводимое под переменные символьного типа (строковые переменные)

Для записи информации на магнитную ленту в интерпретаторе

БЕЙСИК 2.0 ПК-01 используются директивы CSAVE, BSAVE, SAVE, а для ввода информации с магнитной ленты-директивы CLOAD, BLOAD, LOAD, MERGE. Способ записи информации, использующийся указанными директивами, совместим со способом записи применяющимся в персональных компьютерах серии MSX.

Директива CSAVE позволяет сохранить на магнитной ленте БЕЙСИК - программу во внутреннем представлении.

Формат директивы:

CSAVE "⟨имя файла⟩"

где ⟨имя файла⟩ - это последовательность от одного до шести символов. Не рекомендуется в качестве имени файла использовать все пробелы (см. директиву CLOAD).

Директива CLOAD дает возможность загрузить с ленты в память ЭВМ программу, записанную при помощи директивы CSAVE.

Формат директивы:

CLOAD ["[⟨имя файла⟩] "]

Параметр ⟨имя файла⟩ записывается так-же, как и в директиве CSAVE.

Если ⟨имя файла⟩ в директиве присутствует, то поиск на ленте будет вестись до тех пор, пока не будет обнаружен файл с указанным именем. Если же параметр в директиве CLOAD отсутствует, или заданы только кавычки, или в качестве имени файла используются пробелы, то будет прочитан первый по порядку файл, записанный с помощью директивы CSAVE. Прекратить поиск файла можно путем нажатия на клавишу " ⭳ ". Сказанное выше относится и к поиску файлов в директивах BLOAD, LOAD и MERGE за исключением того, что ведется поиск BSAVE - файлов или SAVE -файлов.

По директиве SAVE текст БЕЙСИК-программы записывается на магнитную ленту во внешнем представлении (в коде КОИ-7). Запись ведется блоками по 256 байт. Такую программу можно ввести и выполнить (если

совпадут форматы операторов и функций) на других компьютерах, использующих формат записи MSX ("Корвет", "Ямаха" и др.).

Формат директивы:

SAVE "⟨имя файла⟩ "

Правила записи параметра ⟨имя файла⟩ такие же, как и в директиве CSAVE.

Директива LOAD предписывает загрузить в память программу, записанную директивой SAVE.

Формат директивы:

LOAD [" [⟨имя файла⟩] "]

Директива MERGE дает возможность объединять две программы: программу в памяти с программой, вводимой с магнитной ленты; вторая должна быть записана на ленту директивой SAVE. Если в программе, находящейся в памяти, и в программе, вводимой с магнитной ленты, будут встречаться операторы с одинаковыми номерами строки, то они будут замещаться теми операторами, что вводятся с ленты.

Формат директивы:

MERGE [" [⟨имя файла⟩] "]

Директива BSAVE позволяет записать на магнитную ленту двоичный образ памяти. Это может быть программа в машинном коде или какие - либо данные.

Формат директивы:

BSAVE " ⟨имя файла⟩ ", ⟨нач.адрес⟩ ,⟨кон.адрес⟩ [, ⟨адрес.запуска⟩]

где ⟨имя файла⟩ - см.директиву CSAVE;

⟨нач.адрес⟩ и ⟨кон.адрес⟩ - это адреса соответственно начала и конца области памяти, которая выводится на ленту;

⟨адр.запуска⟩ - точка входа в программу в машинном коде. Если этот параметр опущен, то точка запуска устанавливается равной параметру ⟨нач. адрес⟩ .

Параметры ⟨нач.адрес⟩ , ⟨кон.адрес⟩ и ⟨адр.запуска⟩ должны быть числовыми выражениями.

Директива BLOAD предназначена для загрузки в память информации записанной директивой BSAVE.

Формат директивы:

BLOAD "[⟨имя файла⟩]" [,R] [, ⟨смещение⟩]

где ⟨имя файла⟩ - см директиву CLOAD ;

R - необязательный параметр указывающий, что после загрузки управление должно быть передано на пусковой адрес загруженной программы в кодах машины (автозапуск). Если эта программа завершается командой возврата (RET), то управление возвращается интерпретатору;

⟨смещение⟩ - необязательный параметр, который позволяет загрузить данные в область памяти, отличающуюся от той, из которой они записывались на ленту. Значение параметра ⟨смещение⟩ прибавляется (по модулю 2^{16}) к адресам начала и конца области данных, которые были заданы в директиве BSAVE и сохранены на ленте вместе с данными.

Директивы BSAVE и BLOAD, так же, как и директива RUN , могут использоваться в качестве операторов.

Для загрузки с ленты БЕЙСИК - программы, созданной в предыдущей версии интерпретатора предусмотрена директива SLOAD. Ее формат:

SLOAD

13.2. Диагностические сообщения интерпретатора

В состав интерпретатора включена система контроля, позволяющая выявить большинство ошибок, связанных с недопустимым использованием конструкций БЕЙСИКА. При обнаружении ошибки интерпретатор выдает диагностическое сообщение, которое в случае интерпретации директивы и в режиме непосредственного исполнения содержит код ошибки, а при выполнении БЕЙСИК-программы содержит код ошибки и номер строки оператора, в котором произошла ошибка. Коды ошибок, выдаваемые интерпретатором, приведены в таблице 3.

Таблица 3

Код ошибки	Значение	Пояснение
1	2	3
NF	NEXT без FOR	Программа не содержит соответствующего FOR для NEXT.
SN	Синтаксис	Неправильное использование ограничителей, символов и т.д.
RG	RETURN без GOSUB	Оператор RETURN встретился прежде, чем был использован оператор GOSUB.
OD	Выход за пределы DATA	В программе исчерпаны все элементы списка DATA, а оператор READ пытается еще прочитать данные.
FC	Недопустимый вызов	Параметры, передаваемые функции, несоответствуют их пределам. Возможные причины: 1. Отрицательный индекс в элементе массива. 2. A∧B, где A - отрицательное, а B не является целым. 3. Недопустимый аргумент при обращении к функциям LOG , SQR, MID¤, LEFT¤, RIGHT¤, TAB , SPC. 4. Недопустимые аргументы операторов POKE, PLOT, DRAW , ON .
OV	Переполнение	Значение выражения превышает $1,7 \cdot 10^{38}$
OM	Превышение памяти	Программа слишком большая, содержит очень много циклов, подпрограмм, переменных или сложных выражений.

Продолжение табл. 3

I	2	3
US	Неопределенный номер	Не существует строки, указанной в ссылке.
BS	Превышение границ	Обращение к переменной с индексами, которая вне описанной размерности массива, или в операторе ON выход за пределы списка меток, или неправильное число размерностей в переменной с индексами.
DD	Повторное определение	Повторное описание массива.
/Ø	Деление на Ø	Была предпринята попытка деления на Ø.
ID	Недопустимое непосредственное исполнение	В режиме непосредственного исполнения оператор использован неправильно.
TM	Смешивание типов	Для функции был задан неправильный тип данных: любое неправильное смешивание типов данных.
OS	Превышение памяти для строк	Для переменных символьного типа требуется больше памяти, чем назначено. Можно увеличить память с помощью директивы CLEAR.
LS	Недопустимая длина строки	Длина строки больше 255 символов.
ST	Сложная строка	Строка очень длинная или сложная; ее следует разделить на две части.
CN	Не может быть CONT	Была произведена попытка продолжить программу, которая не допускает продолжения (ни после останова по STOP, ни после

Продолжение табл. 3

I	2	3
		останова по END, ни после останова по функциональной клавиши FØ, или при отсутствии следующих операторов программы).
UF	Неопределенная функция пользователя	Ссылка на неопределенную функцию пользователя.

13.3. Средства редактирования

Средства редактирования интерпретатора позволяют модифицировать программу, исправлять ошибки, допущенные при наборе программы.

Если при выводе строки последний набранный символ неверный, то его можно уничтожить путем нажатия на клавишу ЗБ; для уничтожения двух последних символов необходимо дважды нажать на клавишу ЗБ и т.д.

Для уничтожения всей набранной строки необходимо нажать на функциональную клавишу ⟨F5/AS⟩.

Указанными средствами можно пользоваться до окончания ввода строки, т.е. до нажатия на клавишу ВК.

С целью облегчения ввода наиболее часто встречающихся ключевых слов БЕЙСИКа в интерпретаторе версии 2.0 предусмотрено использование некоторых функциональных клавиш, специальных клавиш и комбинации клавиш СУ и алфавитно-цифровых клавиш. Так, нажатие функциональной клавиши ⟨F1/[illegible]⟩ вызывает ввод ключевого слова CLOAD, функциональ-

ной клавиши ⟨F2/ЗАП⟩ - CSAVE, функциональной клавиши ⟨F3/BS⟩- LIST, функциональной клавиши ⟨F4/ED⟩- EDIT; путем нажатия на клавишу TAB инициируем ввод символов TAB (, а нажатием на клавишу ПС - ввод директивы RUN и символа ВК, т.е. запуск программы на выполнение.

Одновременное нажатие на клавишу СУ и одну из алфавитно-цифровых клавиш вызывает ввод еще одного из 32 ключевых слов БЕЙСИКа (см. табл. 4). Например, комбинация клавиш СУ и С дает слово COLOR, СУ и P - PRINT, СУ и L - LOCATE, СУ и Т - THENи т.д. В ряде случаев первая буква ключевого слова и алфавитно-цифровой символ не совпадают, однако можно составить некоторые мнемонические правила:

СУ и Z - SOUND (звук), СУ и X - PSET, СУ и У - LINE (графика, которая связана с системой координат ХУ) и т.п.

Таблица 4

Таблица ключевых слов

Клавиша	Ключевое слово	Примечание
@	READ	
A	DATA	
B	BSAVE	
C	COLOR	
D	DELETE	
E	RENUM	Не реализовано
F	FOR	
G	GOTO	
H	HEX¤	
I	INPUT	
J	BEEP	
K	POKE	
L	LOCATE	
M	MERGE	
N	NEXT	
C	BLOAD	
P	PRINT	
Q	RESTORE	
R	RUN	
S	STEP	
T	THEN	
U	RETURN	
V	VAL	
W	DRAW	
X	PSET	
У	LINE	
Z	SOUND	
[	LOAD	
\	GOSUB	
]	SAVE	
^	CLEAR	
-	CLS	

Продолжение табл. 4

Для корректировки (замены или исправления) ранее введенной строки необходимо набрать номер корректируемой строки и новое ее содержание, - старое содержимое заменяется новым.

Более мощными средствами редактирования являются средства, предоставляемые директивной EDIT. Формат директивы:

EDIT ⟨номер строки⟩

После ввода этой директивы оператор (операторы) с указанным

номером строки вызываются в специальный буфер редактирования, содержимое которого "высвечивается" в нижней части экрана. При помощи клавиш управления курсором, курсор устанавливается в нужное место буфера, путем нажатия на алфавитно-цифровые клавиши осуществляется замена (исправление) текста редактируемого оператора, клавишей ГТ освобождается место для вставки, клавишей ЗБ уничтожаются ненужные символы. Выход из режима редактирования осуществляется путем нажатия на клавишу ВК; при этом курсор может находиться в произвольной части буфера.

Для удаления какой либо из ранее набранных строк, достаточно набрать номер удаляемой строки и нажать клавишу ВК, - строка с указанным номером удаляется.

Для удаления группы строк программы можно использовать директиву:

DELETE ⟨начальная строка⟩ -⟨конечная строка⟩

где ⟨начальная строка⟩ и ⟨конечная строка⟩ - это номера строк операторов того участка программы, который подлежит удалению.

ПРИЛОЖЕНИЕ

ДОПОЛНИТЕЛЬНЫЕ ОПЕРАТОРЫ БЕЙСИКА

К дополнительным операторам БЕЙСИКа версии 2.0 относятся операторы CIRCLE, PAINT и оператор DEF USR.

В графических операторах CIRCLE, PAINT координаты точек задаются так же, как и в операторах PSET, PRESET и LINE, т.е. точка с координатами (Ø,Ø) находится в левом верхнем углу экрана.

Оператор CIRCLE позволяет выводить на экран видеотерминала окружности, овалы, сектор круга, дуги.

Формат оператора:

⟨номер строки⟩ CIRCLE (⟨X - коорд⟩ ,⟨У - коорд⟩),⟨радиус⟩
[, [⟨цвет⟩] [,[⟨нач.угол⟩] [,[⟨кон.угол⟩] [, ⟨отношение⟩]]]]

где X - коорд и У - коорд - выражения, задающие значения соответственно координаты X и У центра окружности;

⟨радиус⟩ - значение радиуса окружности (дуги окружности);

⟨цвет⟩ - номер цвета переднего плана, которым чертится окружность (дуга);

⟨нач.угол⟩- начало дуги (в радианах); по умолчанию равно нулю;

⟨кон.угол⟩- конец дуги (в радианах); по умолчанию равно 2π;

⟨отношение⟩ - характеристический коэффициент, задающий отношение горизонтальной и вертикальной осей и позволяющий рисовать овалы; по умолчанию равен I.

Если заданы параметры ⟨нач.угол⟩, ⟨кон.угол⟩, то рисуется дуга окружности. Если любой из этих параметров отрицательный, то используется его абсолютное значение; при этом соответствующий конец дуги соединяется с центром линии радиуса. Значение углов должно находиться в пределах от Ø до 6.28318.

При указании параметра ⟨отношение⟩ рисуется овал, у которого отношение вертикальной оси к горизонтальной равно значению параметра.

Если значение параметра меньше I, то размер горизонтальной оси равен диаметру (двум значениям параметра ⟨радиус⟩), а при значении большем I - вертикальная ось равна диаметру.

Оператор PAINT используется для закрашивания области экрана видеотерминала сплошным цветом.

Формат оператора:

⟨номер строки PAINT (⟨X - коорд⟩, ⟨У - коорд⟩), ⟨цвет⟩ [,⟨граница⟩]

где ⟨X - коорд⟩, ⟨У - коорд⟩ - выражения, значения которых равны соответственно координате X и координате У - точки, с которой начинается закрашивание области;

⟨цвет⟩ - выражение, задающее значение номера цвета закраски (если опущено, то используется цвет переднего плана, определенный ранее);

⟨граница⟩ - выражение, определяющее номер цвета границы области закраски (если опущено, то номер цвета границы совпадает с номером цвета закраски).

Параметр ⟨граница⟩ указывается в том случае, если цвет закрашивания не совпадает с цветом границы закрашиваемой области.
Точка начала закраски области может быть любой точкой внутри области (но не на границе). Оператор PAINT проверяет наличие границ только по вертикали и горизонтали - и объект признается ограниченным, если ограничены вертикаль и горизонталь. Но малейшая прореха в границе позволяет оператору PAINT "пролезть" наружу и там закрасить область экрана (в том числе и всего).

Следующий пример иллюстрирует применение операторов CIRCLE и PAINT:

```
IØ  CLS
2Ø  CIRCLE (IØØ, IØØ), 9Ø, I,,,3
3Ø  CIRCLE (IØØ, IØØ), 7Ø, ,,,Ø.5
4Ø  LINE (IØØ, 4Ø) - (IØØ, I6Ø),Ø
5Ø  LINE (4Ø, IØØ) - (I6Ø, IØØ)
```



```
6Ø  PAINT (IØØ, IØØ), 3,I
```

Оператор DEF USR определяет адрес точки входа в определенную пользователем подпрограмму в машинном коде.

Формат оператора:

⟨номер строки⟩ DEF USR = ⟨выражение⟩

где ⟨выражение⟩ - это арифметическое выражение определяющее адрес точки входа в подпрограмму.

Параметр ⟨выражение⟩ должен иметь положительное значение в диапазоне от Ø до 65535; при нецелом значении дробная часть отбрасывается, а при значении большем чем 65535 берется остаток от деления на 65536. Полученное значение адреса (2 байта) заносится в ячейки памяти с адресом 73 или 74. Определенный оператором DEF USR адрес действительный только во время выполнения БЕЙСИК-программы (или программной строки в режиме непосредственного исполнения) и при выходе в режим диалога стирается.

Напомним, что запуск подпрограммы в машинном коде на выполнение осуществляется с помощью функции USR.

Тип. ЛПО им. Б. И. Ленина. Зак. 2077-1500